REMARKS

Claims 1-34 remain pending. Favorable reconsideration of the application is requested in view of the following remarks. No new matter has been added.

Claims 1-32 and 34 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Zhang et al (US 6,282,699) in view of Hartmut Poglheim (Genetic and Evolutionary Algorithm Toolbox for use with Matlab). Claim 33 stands rejected over Zhang in view of Poglheim and in further view of David M. Gay (Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming).

Applicants respectfully disagree that the cited references teach or suggest every limitation of the rejected claims.

Independent claim 1 recites a method for invoking a method defined within an object-oriented environment. The method entails, among other things, retrieving a set of method signatures, where each signature corresponds to a method provided by an object within the object oriented environment, and wherein each signature includes a method name and lists any data types received by the corresponding method. The method signatures are compared to data types of input parameters received from an array-based computing environment, and then ranked as a function of the comparison. This ranking is used to select one of the signatures according to the ranking, and the method corresponding to the selected signature is then invoked. Thus the claimed method permits, for example, program code in an array-based computing environment to invoke methods within an object-oriented environment without having to explicitly select from among methods having the same names but differing method signatures. The method may be used to automatically select from among the same named methods according to how well each method matches the data types of the parameters of the method invocation.

Nothing in Zhang or Polgheim teaches the invention of claim 1. Zhang discloses an extension to a graphical programming system called a "code node" that enables a programmer of the graphical programming system to invoke execution of textual code. The code node may be connected to other nodes in the graphical programming system and receive input data, which is used during execution of the textual code. The result of executing the textual code is returned to the code node as an output of the node to the graphical program. (abstract).

Contrary to the requirement of claim 1, nothing in Zhang teaches or suggests a method for selecting among methods in an object-oriented environment, based on a comparison of method signatures with the data types of input parameters. First, there is simply no teaching or suggestion in Zhang of selecting from among methods in an object oriented environment. Rather, in Zhang, a particular piece of code is associated by the programmer with the code node and it is that code that is executed at run time (col. 11, ln 34; col. 12, ln 33). Moreover, all of the parameters (terminals) that are passed to the textual code have data types that are explicitly defined by the programmer from a list of options provided by the system prior to run time (col. 19, ll. 8-15). If the user does not configure the correct data type, an error occurs (col. 19, ll. 16-18). Thus, Zhang teaches away from using the data types of input parameters to select from among method signatures of methods in an object oriented environment.

Nor is there any teaching or suggestion in Zhang of selecting the correct method in an object orient environment to be invoked based on a comparison of a method signature with input parameters. Method signatures are never described or suggested in Zhang, nor are any such comparisons.

The examiner states that "a set of method signatures" is described in Zhang at col. 13, ln. 40-53. However careful review of that passage reveals no such description. The cited passage of Zhang states that "when a specific code node or script node is selected from the function palette by the user, the graphical programming system receives the object class and also receives an index into the resource string. From this index, the graphical programming system obtains the script server identification and the name of the DLL to load. Handles to the DLL are stored in a global array (FIG. 7)." Nothing in this passage teaches or suggests a method signature. Rather, this passage merely describes that when a programmer inserts a code node or script node into a graphical program, the appropriate DLL for executing the code associated with the code node or script node is loaded into the system. There is no teaching or suggestion of <u>any</u> technique for selecting from among multiple methods signatures, let alone the complete technique recited in claim 1.

Nor does Poglheim teach or suggest modifying Zhang to provide the missing claim limitations. Poglheim describes a MATLAB toolbox for implementing genetic and evolutionary algorithms. These are abstract stochastic search methods that mimic biological evolution to

determine the best match to a given set of search criteria. Nothing in Poglheim teaches or suggests genetic or evolutionary algorithms should be used in selecting from among methods in an object-oriented environment based on a comparison of method signatures with the data types of input parameters.

For at least the foregoing reasons, claim 1 is patentable over Zhang in view of Poglheim. For similar reasons, claims 12 and 23 or also patentable over Zhang in view of Poglheim. Claim 12 recites a computer program for performing substantially the same steps of claim 1. Claim 23 recites a system including a signature selector that retrieves and ranks signatures corresponding to methods defined within an object-oriented environment, a structure that is nether taught nor suggested by Zhang or Polgheim.

The remaining claims 2-11, 13-22, and 24-34 are all dependent from one of base claims 1, 12 or 23, and are therefore patentable for at least the same reasons as the base claims.